Cutting stock problem with usable leftovers: an approach using K-best solutions

Arthur Medeiros Figueiredo Barreto
Departamento de Engenharia de produçao - UNESP
arthur.medeiros@unesp.br

Douglas Nogueira do Nascimento
Departamento de Engenharia de produçao - UNESP
douglasnn@fc.unesp.br

Luiz Henrique Cherri
ODM - Optimized Decision Making
lhcherri@icmc.usp.br

Adriana Cristina Cherri
Departamento de Matemtica - UNESP
adriana@fc.unesp.br

Abstract

This paper addresses the one-dimensional cutting stock problem with usable leftovers. In this problem, cutting patterns can generate leftovers that can be used in the future to meet new demands. To solve this problem, modifications were proposed in the simplex method with column generation, that is one of the most used methods to solve cutting stock problems. Basically, this modification consists of generating K-best cutting patterns in each iteration of the simplex method while searching for the best solution. Although previous works use k-best solutions for the knapsack problem, to the best of our knowledge, there is no paper in the literature that extends it to the cutting stock problems with usable leftovers. Computational experiments were performed with instances from the literature solved by the traditional simplex method with column generation and the results showed that the iteration counter and the computational effort are reduced.

Keywords: Cutting stock problem; Usable leftovers; Cutting and packing; Solution pool; Knapsack problem.

1 Introduction

In the industries, the sector that deals with cutting problems constantly face with the problem of defining the way in which the objects available in stock must be cut in order to meet the demand of clients. Generally, this decision-making process is complicated since there are numerous ways to cut each object in smaller items.

The definition of the one-dimensional cutting stock problem with usable leftovers (CSPUL) is similar to the definition of the classic cutting stock problem. The difference between both is that the cutting patterns in the CSPUL can generate leftovers that return to stock to meet future demands and thus they are not accounted as waste. Numerous studies were carried out looking for good solutions for cutting stock problems (CSP), but few are related to the CSPUL.

[10] proposed a mathematical model to represent the CSPUL. The solution method consists of including additional items to be produced (future leftovers) without a demand to be met. To solve this problem the column generation method (Gilmore and Gomory, 1963) was used. To solve the CSPUL in a garb industry, [5] proposed a mathematical model with the objective of minimizing the waste and the number of items not produced. The authors also proposed a heuristic procedure to solve the problem.

Based on heuristics from the literature, [2] proposed solution methods to solve the CSPUL. [4] proposed an extension of the study of [10], diversifying the types of objects available in stock and limiting the quantity of leftovers that could be generated during the cutting process. In [3], a survey addressing to studies about CSPUL was developed, presenting the possible applications, mathematical models and comments referring the results found by these models.

A mathematical model to represent the CSPUL was proposed by [1]. In this model, the leftovers are treated explicitly and have their length and quantity previously defined. To solve this problem, the authors relaxed the integrality condition of the model variables and use the simplex method with column generation [6] to solve it.

Solving the CSP using the simplex method with column generation consists of dividing the problem in two parts, (1) the master problem, (2) and the subproblem. The master problem is the cutting problem stock problem solved using a reduced number of patterns. The subproblem is the *knapsack problem*, which is used to generate the most attractive cutting patterns to the master problem. At each iteration of the method, a new column is generated by the subproblem and included in the master problem to improve its solution quality. With this strategy to solve the problem, the number of iteration and the computational time increase as the dimension of the problem increases.

To reduce the quantity of iterations and consequently the computational time, one can select more than one cutting pattern with potential to compose the solution in each iteration of the method. [12] proposed a column generation method based on dynamic programming to determine the K-best solutions to the knapsack problem. [7] used the best K-solutions for the knapsack problem in heuristics based on the column generation to solve the compartmentalized knapsack problem. [8] determine the k-best solutions to the knapsack problem using the branch and bound method and pointed out its numerous applications.

In this paper, we use the mathematical model proposed in [1] to represent the CSPUL and, to solve it, the column generation method is used considering the strategy of generating the k-best solutions of knapsack problem. Computational experiments were performed using instances from the literature. Our main purpose is to decrease the computational time and iterations number from the original strategy. The k-best cutting patterns were obtained using the CPLEX solver.

After this introductory section, Section 2 presents the definition of the CSPUL and the mathematical model proposed by [1]. In Section 3, the proposed method is presented. In Section 4, the results of the computational experiments are shown. Finally, Section 5 concludes the paper.

2 Problem Description

The problem presented in [1] consists of producing a set of demanded items by cutting standard objects (bought on the market) and leftovers (from previously cutting processes) available in stock. The demands must be met by cutting available objects and leftovers that are available in stock with the objective of minimizing the waste. Leftovers with predefined lengths can be generated in limited quantity. Note that these leftovers are not accounted as a waste.

The parameters and variables used in the model are defined as:

- S: number of types of standard objects. We denote object type $s, s \in \{1, ..., S\}$;
- R: number of types of leftovers in stock. We denoted leftover type $S + s, s \in \{1, ..., R\}$;
- e_s: number of objects/leftover type s available in stock s, $s = 1, ..., S + R$;
- m: number of types of ordered items;
- d_i: demand for item type $i, i = 1, ..., m$;
- J_s: set of cutting patterns for object type $s, s = 1, ..., S + R$;
- $J_s(k)$: set of cutting patterns for standard object type s with leftover type $S + k, k = 1, ..., R, s = 1, ..., S$;
- a_{ijs}: number of items type i in cutting pattern j for object type $s, i = 1, ..., m, s = 1, ..., S + R, j \in J_s$;
- a_{ijsk}: number of items type i in cutting pattern j for object type s and leftover type $S+k, i = 1, ..., m, k = 1, ..., R, s = 1, ..., S, j \in J_s(k)$;
- c_{js}: waste of cutting the object/leftover s according to pattern $j, s = 1, ..., S + R, j \in J_s$;
- c_{jsk}: waste of cutting object s according to pattern j when generating a leftover type $S+k, s = 1, ..., S, k = 1, ..., R, j \in J_s(k)$;
- U: maximum number of leftovers;

Variables:

- x_{js}: number of objects type s cut according to pattern j, $s = 1..., S + R, j \in J_s$;
- x_{jsk}: number of objects type s cut according to pattern j generating a leftover type $S+k, s = 1, ..., S, k = 1, , R, j \in J_s(k)$.

The mathematical model proposed by the authors is given by:

$$\min \text{ f(x)} = \sum_{s=1}^{S}\sum_{j\in J_s} c_{js}x_{js} + \alpha' \sum_{s=1}^{S}\sum_{k=1}^{R}\sum_{j\in J_s(k)} c_{js}x_{js} + \alpha'' \sum_{s=S+1}^{S+R}\sum_{j\in J_s} c_{js}x_{js} \quad (1)$$

$$\text{subject to} \quad \sum_{s=1}^{S}\sum_{j\in J_s} a_{ijs}x_{js} + \sum_{s=1}^{S}\sum_{k=1}^{R}\sum_{j\in J_s(k)} a_{ijsk}x_{jsk} + \sum_{s=S+1}^{S+R}\sum_{j\in J_s} a_{ijs}x_{js} = d_i, \qquad i = 1, ..., m \quad (2)$$

$$\sum_{j\in J_s} + \sum_{k=1}^{R}\sum_{j\in J_s} x_{jsk} \leq e_s, \qquad s = 1, .., S \quad (3)$$

$$\sum_{j\in J_s} x_{js} \leq e_s, \qquad s = S+1, ..., S+R \quad (4)$$

$$\sum_{s=1}^{S}\sum_{k=1}^{R}\sum_{j\in J_s(k)} x_{jsk} - \sum_{s=S+1}^{S+R}\sum_{j\in J_s} x_{js} \leq U - \sum_{s=S+1}^{S+R} e_s \quad (5)$$

$$x_{js} \geq 0, s = 1, ..., S+R, j \in J_s$$

$$x_{jsk} \geq 0, k = 1, ..., R, s = 1, ..., S, j \in j_s(\text{k}) \text{ and integer.} \quad (6)$$

In the model (1) (6), the objective function minimizes total waste when cutting standard objects (even when leftovers are generated) and leftovers available in stock. Constraint (2) guarantees the fulfillment of demand. Constraints (3) and (4) guarantees that the quantity of objects to be cut do not be greater than the quantity available in stock. Constraint (5) limits the quantity of leftovers that can be generated. Constraint (6) ensures the non-negativity and integrality of the decision variables.

Due to the exponential number of variables and the integrality conditions (constraint (6)), it is very difficult to find the optimal solution for the model in the model (1) (6). Therefore, these conditions were relaxed and continuous solutions for the CSPUL could be found using simplex method with column generation.

Considering the variety of objects in stock the $k-$best solutions to the simplex method with column generation was used to found the solution to this problem. This strategy is described in the next section.

3 Solution Method

To solve the mathematical model (1)-(6), [1] relaxed the integrality condition of the model and use the simplex method with column generation ([6]). This strategy consists of dividing the problem into a master problem, that is solved considering a reduced number of attractive cutting patterns, and a subproblem, used to generate the attractive cutting patterns to the master problem. Starting with a set of cutting patterns, the master problem is solved. The dual variables related to the

solution of the master problem point to the direction in which more attractive columns could be found. These variables are used to compose the objective of the subproblem (*knapsack problem* supporting in the search for a more attractive cutting pattern. This cutting pattern is inserted in the master problem and the cycle is repeated until it is not possible to make better the quality of the solution in the master problem.

To use the $k-$best solutions ($k-$best cutting patterns) in the column generation method to solve the CSPUL, modification should be considered in subproblem solving phase. Specifically, instead of select the best cutting pattern generated by the subproblem to be inserted in the master problem, the $k-$best cutting patterns are selected and simultaneously inserted in the master problem. Since it was defined, the k value do not change during the procedure.

Since the CSPUL has a great diversity of object types in stock, one subproblem will be solved for each type and, consequently, the k-patterns with smaller relative costs will be selected for each object. These patterns are organized according to non increasing order of the smaller relative costs (in the total, there are $k \times (2 \times S + R)$ selected). Next, the $k-$best cutting patters, with the smallest relative cost are selected and inserted in the master problem.

To select and store the $k-$best columns found by the subproblem, the *solution pool* (a tool provided by CPLEX to retain answers) was used with the commands "solve" and "populate". In order to get a bigger diversity of results, the method "populate" was used to get more answers as it explores the branch and cut tree previously created.

Using this strategy to solve the problem, a continuous solution is found. To obtain an integer solution a heuristic procedure is necessary. However, as the solution obtained with the strategy presented will be compared with the continuous solutions in [1], the heuristic procedure was omitted here.

4 Computational results

To evaluate the performance of the proposed approach, the model (1) - (6) was coded in C++ programming language using CPLEX software, version 12.8. To run the computational experiments, a computer Intel Core i5, 2.7 GHz, 8GB RAM was used.

Some instances presented in [1] were used to evaluate the performance of the proposed method. For the experiments, 6 classes of instances were considered, and in each class, 50 instances were randomly generated by varying the length and demand of items. The instances considering one type of object in stock with length $L_1 = 1000$ and availability enough to meet the demand. In the instances, we considered that in the beginning of the process there is no leftover in stock. However, three types of leftovers can be generated during the cutting process with lengths equal to 400, 500 or 600. Five possibilities for the maximum number of leftovers (U) were evaluated, specifically 0, 3, 6, 9 and 12.

Each class has 15 types of items to be produced that were classified as *medium* (M), if the length was generated in the interval [140, 400] and *big* (B), if the length was generated in the interval [300, 400]. The demands are classified as *small* (S), if they were generated in the interval [1, 10], *medium* (M), if they were in the interval [10, 50] and *big* (B), if they were generated in the interval [50, 300]. As in [1], a class [B,S] represents a set of 50 instances where the items are big, and demands are small.

The values used for k were 1, 2, 4, 6 and 8. Notice that setting k-value as 1, the solution is the same as in [1].

In tables (1) - (6) the number of iterations and time spent when varying the k-value are presented. Column '"It" represents the number of iterations needed to achieve the optimal solution while the column "Time" shows the computational time in seconds.

Table 1: Class [B, B]

	$U = 0$		$U = 3$		$U = 6$		$U = 9$		$U = 12$	
	It	Time	It	Time	It	Time	It	Time	It	Time
$k = 1$	13.98	0.96	13.14	1.58	13.60	1.50	13.16	1.52	13.16	1.62
$k = 2$	7.20	**0.86**	7.84	**1.36**	7.84	**1.40**	7.84	**1.30**	7.86	**1.34**
$k = 4$	4.54	1.06	4.84	1.78	4.84	2.10	4.86	1.88	4.86	2.00
$k = 6$	3.74	1.18	4.00	1.74	4.02	1.76	4.02	1.78	4.02	1.92
$k = 8$	3.18	1.18	3.38	1.62	3.38	1.72	3.38	1.64	3.38	1.72

Table 2: Class [B, M]

	$U = 0$		$U = 3$		$U = 6$		$U = 9$		$U = 12$	
	It	Time	It	Time	It	Time	It	Time	It	Time
$k = 1$	13.02	0.86	12.68	1.80	12.80	1.90	12.78	**1.58**	12.96	2.00
$k = 2$	6.80	**0.80**	7.30	**1.64**	7.30	**1.66**	7.36	1.70	7.42	**1.44**
$k = 4$	4.22	1.06	4.54	1.96	4.54	1.74	4.58	1.74	4.62	1.92
$k = 6$	3.22	1.60	3.48	1.70	3.52	1.80	3.58	1.86	3.60	1.72
$k = 8$	2.84	1.44	3.06	1.84	3.06	1.92	3.08	2.08	3.10	1.80

Table 3: Class [B, S]

	$U = 0$		$U = 3$		$U = 6$		$U = 9$		$U = 12$	
	It	Time	It	Time	It	Time	It	Time	It	Time
$k = 1$	13.34	**1.12**	13.6	**2.14**	14.16	2.68	14.56	2.18	14.92	2.26
$k = 2$	7.54	1.28	8.18	2.18	8.32	**2.04**	8.72	**1.82**	8.98	**2.18**
$k = 4$	4.58	1.58	5.18	2.54	5.26	2.48	5.48	2.50	5.76	2.26
$k = 6$	3.74	1.42	4.24	2.74	4.52	2.66	4.52	2.36	4.76	2.60
$k = 8$	3.30	1.64	3.74	2.72	3.86	2.38	3.98	2.72	4.06	2.76

Table 4: Class [M, B]

	$U = 0$		$U = 3$		$U = 6$		$U = 9$		$U = 12$	
	It	Time	It	Time	It	Time	It	Time	It	Time
$k = 1$	32.72	3.42	31.94	**16.34**	31.86	**17.60**	31.74	**15.86**	31.70	19.10
$k = 2$	19.70	**3.04**	20.00	20.32	20.00	19.82	20.00	18.44	19.96	**17.82**
$k = 4$	13.00	3.72	13.30	23.44	13.24	22.08	13.24	20.12	13.14	21.96
$k = 6$	10.44	4.54	10.60	25.04	10.68	27.74	10.62	23.16	10.56	25.46
$k = 8$	9.46	5.52	9.42	26.80	9.54	28.72	9.46	27.26	9.48	26.16

For all the classes of instances, it is possible to observe a great reduction in the number of iterations when the value of k is increased. This behavior was expected since the with a larger k more attractive columns are inserted in the master problem, making a wider step in the direction of proof the solution optimality. Related to the computational times, in several cases their are very

Table 5: Class [M, M]

	$U = 0$		$U = 3$		$U = 6$		$U = 9$		$U = 12$	
	It	Time	It	Time	It	Time	It	Time	It	Time
$k = 1$	32.90	3.58	32.82	**19.12**	32.62	**17.28**	32.70	18.32	32.72	21.38
$k = 2$	20.26	3.12	20.32	20.64	20.04	18.24	20.04	**18.02**	20.10	**18.64**
$k = 4$	13.00	**3.08**	13.48	24.68	13.16	29.42	13.30	21.60	13.12	21.84
$k = 6$	10.60	4.16	10.94	25.34	10.86	30.74	10.90	27.70	10.84	24.94
$k = 8$	9.16	6.86	9.32	27.86	9.40	30.86	9.22	28.84	9.26	26.78

Table 6: Class [M, S]

	$U = 0$		$U = 3$		$U = 6$		$U = 9$		$U = 12$	
	It	Time	It	Time	It	Time	It	Time	It	Time
$k = 1$	33.56	4.26	34.50	**20.24**	34.82	**19.90**	34.98	**20.96**	34.98	**19.84**
$k = 2$	20.54	**3.98**	20.88	21.40	21.18	21.96	21.24	21.02	21.24	21.44
$k = 4$	14.00	4.18	14.28	23.94	14.20	23.06	14.22	24.06	14.20	24.18
$k = 6$	11.48	5.70	11.72	26.84	11.72	27.42	11.72	27.50	11.70	28.18
$k = 8$	10.56	6.80	10.70	31.08	10.60	29.04	10.46	28.08	10.48	29.04

close and 60% of the classes had smaller computational time when k is larger than one. However, increase the number of columns generated does not imply in the reduction of the computational time since the generation of more cutting patterns at once increment the computational effort of the task.

Concerning the classes, it is possible to see that the demand of the items do not exert a big difference on the computational times or the number of iterations needed to prove the optimality. It happens because the complexity column generation method does not increase with the increase of the items demand. On the other hand, more iterations and time are needed to solve problems with medium items than to solve problems with big items. This characteristic was expected since with smaller items, the number of admissible cutting patterns increases.

5 Conclusions

In this paper, the simplex method with column generation was modified to solve the cutting stock problem with usable leftovers. Specifically, it changes the generation of the cutting patterns, which are included in to the master problem in sets of a predefined size k at each iteration instead of include one cutting pattern at once.

Computational experiments were performed with instances from the literature and for a variety values of k. The results, show a large reduction in the number of iterations to prove the solution optimality in a competitive computational time.

Further analysis must be taken in order to search for the best k-value and more computational experiments must be performed. Also, a heuristic procedure to find integer solutions will be proposed.

6 Acknowledgments

The authors would like to thank the Coordenação de Aperfeioamento de Pessoal de Nivel Superior (CAPES) for the financial support. The authors also are thanks to the authors from [1], who provided the instances for the tests.

References

[1] M. N. Arenales, A. C. Cherri, D. N. Nascimento, A. Vianna. A new mathematical model for the cutting stock/leftover problem *Pesquisa Operacional*, 35(3): 509–522, 2015.

[2] A. C. Cherri, M. N. Arenales and H. H. Yanasse. The one-dimensional cutting stock problem with usable leftoverA heuristic approach. *European Journal of Operational Research*, 196(3):897–908, 2009.

[3] A. C. Cherri, M. N. Arenales, H. H. Yanasse, K. C. Poldi, A. C. Gonçalves. The one-dimensional cutting stock problem with usable leftovers–A survey. *European Journal of Operational Research*, 236(2): 395–402, 2014.

[4] Y. Cui and Y. Yang A heuristic for the one-dimensional cutting stock problem with usable leftover. *European Journal of Operational Research*, 204(2): 245–250, 2010.

[5] M. Gradišar, J. Jesenko and G. Resinovič Optimization of roll cutting in clothing industry. *Computers & Operations Research*, 24(10): 945–953, 1997.

[6] P. C. Gilmore and R. E. Gomory. A linear programming approach to the cutting stock problemPart II. *Operations research*, 11(6): 863–888, 1963.

[7] A. A. S. Leo, M. O. Santos, R. Hoto and M. N. Arenales. . The constrained compartmentalized knapsack problem: mathematical models and solution methods. *European Journal of Operational Research*, 212(3): 455–463, 2011.

[8] A. A. S. Leo, Cherri, L. H. and M. N. Arenales. Determining the K-best solutions of knapsack problems. *Computers & Operations Research*, 49: 71–82, 2014.

[9] K. C. Poldi and M. N. Arenales. Heuristics for the one-dimensional cutting stock problem with limited multiple stock lengths. *Computers & Operations Research*, 36(6): 2074–2081, 2009.

[10] G. Scheithauer. A note on handling residual lengths. *Optimization*, 22(3): 461–466, 1991.

[11] L. Tomat and M. Gradišar. One-dimensional stock cutting: optimization of usable leftovers in consecutive orders. *Central European Journal of Operations Research*, 25(2): 473–489, 2017.

[12] H. H. Yanasse, N. Y. Soma and N. Maculan. An algorithm for determining the k-best solutions of the one-dimensional knapsack problem *Pesquisa Operacional*, 20(1): 117–134, 2000.